J Comput Virol Hack Tech (2016) 12:235–242
DOI 10.1007/s11416-015-0260-0

ORIGINAL PAPER

On normalized compression distance and large malware

Towards a useful definition of normalized compression distance for the classification of large files

Rebecca Schuller Borbely[1]

Received: 3 September 2015 / Accepted: 8 December 2015 / Published online: 30 December 2015

Abstract Normalized Compression Distance (NCD) is a popular tool that uses compression algorithms to cluster and classify data in a wide range of applications. Existing discussions of NCD's theoretical merit rely on certain theoretical properties of compression algorithms. However, we demonstrate that many popular compression algorithms do not seem to satisfy these theoretical properties. We explore the relationship between some of these properties and file size, demonstrate that this theoretical problem is actually a practical problem for classifying malware with large file sizes, and propose some variants of NCD that mitigate this problem.

1 Introduction

In the era of big data, techniques that allow for data understanding without domain expertise enable more rapid knowledge discovery in the sciences and beyond. One technique that holds such promise is the Normalized Compression Distance (NCD) [14], which is a similarity measure that operates on generic file objects, without regard to their format, structure, or semantics.

NCD approximates the Normalized Information Distance, which is universal for a broad class of similarity measures. Specifically, the NCD measures the distance between two files via the extent to which one can be compressed given the other, and can be calculated using standard compression algorithms.

✉ Rebecca Schuller Borbely
rborbely@cyberpointllc.com

[1] CyberPoint International, 621 E. Pratt St., Suite 300, Baltimore, MD 21202, USA

NCD, and its open source implementation CompLearn [5] have been widely applied for clustering, genealogy, and classification in a wide range of application areas. Its creators originally demonstrated its application in genomics, virology, languages, literature, music, character recognition, and astronomy [7]. Subsequent work has applied it to plagiarism detection [4], image distinguishability [19], machine translation evaluation [20], database entity identification [18], detection of internet worms [22], malware phylogeny [21], and malware classification [1] to name a few.

Assuming some simple properties of the compression algorithm used, the NCD has been shown to be, in fact, a similarity metric [7]. However, it remains to be seen whether real word compression algorithms actually satisfy these properties, particularly in the domain of large files. As data storage has become more affordable, large files have become more common, and the ability to analyze them efficiently has become imperative. Music recommendation systems work with MP3s which are typically several megabytes in size, medical images may be up to 30 MB or more [9], and computer programs are often more than 100 MB in size.

This paper explores the relationship between file size and the behavior of NCD, and proposes modifications to NCD to improve its performance on large files; these improvements are demonstrated on two malware classification problems.

Section 2 provides an introduction to NCD and the compression algorithm axioms that have been used for proving it to be a similarity metric. Section 3 explores the extent to which several popular (and not-so-popular) compression algorithms satisfy these axioms and investigates the impact of file size on its effectiveness for malware classification. Finally, Sect. 4 proposes two possible adaptations of the NCD definition, for the purpose of improving its performance on large files, and demonstrates significant performance

improvement with several compressors on two malware classification problems.

2 NCD Background

The motivating idea behind the Normalized Compression Distance (NCD) is that the similarity of two objects can be measured by the ease with which one can be transformed into the other. This notion is captured formally by the *information distance*, $E(X, Y)$, between two strings, X, Y, which is the length of the shortest program that can compute Y from X or X from Y in some fixed programming language. The information distance generalizes the notion of Kolmogorov complexity, where $K(X)$ is the length of the shortest program that computes X, and intuitively captures a very general notion of what it means for two objects to be similar.

However, for the purposes of computing similarity, it is important that distances be relative. Two long strings that differ in a single character should be considered more similar than two short strings that differ in a single character. This leads to the definition of the Normalized Information Distance (NID),

$$\mathrm{NID}(X, Y) \equiv \frac{E(X, Y)}{\max(K(X), K(Y))}$$

The NID has several nice features: it satisfies the conditions of a metric up to a finite additive constant, and it is universal, in the sense that it minorizes every upper semi-computable similarity distance [7]. However, it is also incomputable, which is a serious obstacle.

Given a compression algorithm, C, $E(X, Y)$ can, in some sense, be approximated by $C(XY)$, the result of compressing with C the file consisting of X concatenated with Y, and $\mathrm{NID}(X, Y)$ can, in turn, be approximated by

$$\mathrm{NCD}(X, Y) \equiv \frac{|C(XY)| - \min(|C(X)|, |C(Y)|)}{\max(|C(X)|, |C(Y)|)}$$

However, in order to prove that NCD is a similarity metric, [7] placed several restrictions on the compression algorithm. A compression algorithm satisfying the conditions below is said to be a *normal* compressor.

Normal Compression A normal compressor, C, as defined in definition 3.1 in [7], is one that satisfies the following, up to an additive $O(\log n)$ term, where n is the largest length of an element involved in the (in)equality concerned:

- Idempotence: $|C(XX)| = |C(X)|$ and $|C(\lambda)| = 0$, where λ is the empty string.
- Monotonicity: $|C(XY)| \geq |C(X)|$.
- Symmetry: $|C(XY)| = |C(YX)|$.
- Distributivity: $|C(XY)| + |C(Z)| \leq |C(XZ)| + |C(YZ)|$.

where $C(X)$ denotes the string X' resulting from the application of compressor C to string X, XY denotes the concatenation of X and Y, and $|X|$ denotes the length of string (or file) X.

While using these properties to prove that NCD is a similarity metric is well beyond the scope of this paper, it may be worthwhile to shed some intuition on the role they play. Symmetry and distributivity correspond closely the properties that comprise the definition of a metric. Most simple is the property of symmetry: it makes little sense to talk about the distance between two objects if that distance changes depending which object one starts with. Somewhat less intuitively, the distributivity property is related to the triangle inequality, which essentially says that the shortest distance between two objects is a straight line. The monotonicity property provides for consistent behavior of compression, assuring that if you add more data, the compressed size doesn't decrease. Finally, the idempotence property says simply that if an object comprises a simple duplication of a smaller object, the compression algorithm should be able to take advantage of that, and come close to compressing it to the size to which it can compress the smaller object. (E.g., if a file contains the string "abcdefg.abcdefg.", one should be able to compress it in the spirit of "2*abcdefg.".) While this seems intuitive enough, we will see that idempotence is not so simple to achieve in practice.

The question remains whether existing compression algorithms satisfy these axioms, particularly in the domain of large files. While NCD has apparently been quite successful in practice, the majority of applications (see Sect. 1) have been on relatively small files. Notably, music applications [6,7], used MIDI files rather than the more common, and much larger, MP3 format.

Previous work [3] explored the NCD distance from a file to itself (which is closely related to the idempotence axiom) for bzip, zlib, and PPMZ on the Calgary Corpus [23], comprising 14 files, the largest of which is under 1 MB. The following section explores these axioms on a larger and more representative dataset and investigates the practical impact of deviations from normality.

3 Application of NCD to large files

3.1 Normality of compression algorithms

The definition of a normal compressor deals with asymptotic behavior, allowing for an $O(\log(n))$ discrepancy in the axioms of idempotence, monotonicity, symmetry, and distributivity. Thus, in theory, experimental validation (or

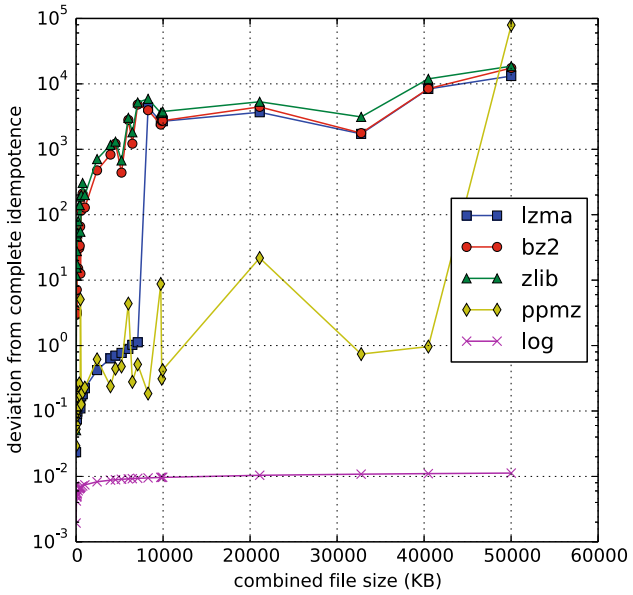

Fig. 1 Idempotence on compression corpora: $|C(XX)| - |C(X)|$ as compared to $\log(|XX|)$ versus $|XX|$

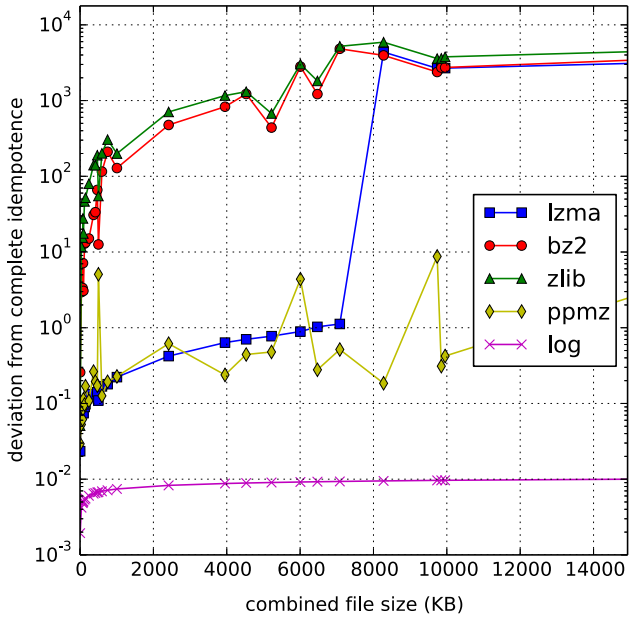

Fig. 2 Idempotence on compression corpora: Enlargement of a portion of the graph in Fig. 1 to more clearly show the behavior for smaller files

refutation) of these axioms is not truly feasible – perhaps the behavior changes when the file size is beyond that of the largest file in our experiment. Nonetheless, we endeavor to experimentally explore these axioms more extensively than has been done in prior work.

Data We combined the traditional Calgary Corpus with the Large and Standard Canterbury Corpora, as well as the Silesia Corpus[1]. The latter contains files of size ranging from 6 MB to 51 MB, greatly expanding the size distribution over the corpus explored in [3].

Idempotence Figures 1 and 2, show the difference in the sizes of $C(X)$ and $C(XX)$, and $\log(|XX|)$, for a representative subset of files X in the dataset, with C ranging over compression algorithms bzip2 [17], lzma [16], PPMZ [2], and zlib [10]. Indeed, bz2 and zlib quite apparently fail the idempotence axiom, with $|C(XX)|$ growing much faster than $|C(X)|$, with a term of $O(\log(|XX|))$ unable to put a dent in the difference. While PPMZ and lzma appear significantly better for smaller file sizes, still, this value grows much faster than $\log(|XX|)$, as apparent in Fig. 2. We see that lzma makes a large jump around 8 MB (but even before that, its growth is much larger than the log function).

Symmetry Figure 3 shows the magnitude of difference between $|C(XY)|$ and $|C(YX)|$. While in most cases, at this scale, this was bounded by $\log(|XY|)$ (and in all cases by a small constant factor thereof), the asymptotic behavior is unclear, as values for all four compressors spike wildly. This is likely due to the fact that the extent of the symmetry is dependent on the compressibility, similarity, and/or size disparity of the two files involved. zlib and lzma look quite promising for symmetry, while the asymptotic behavior of PPMZ and bz2 is not discernible.

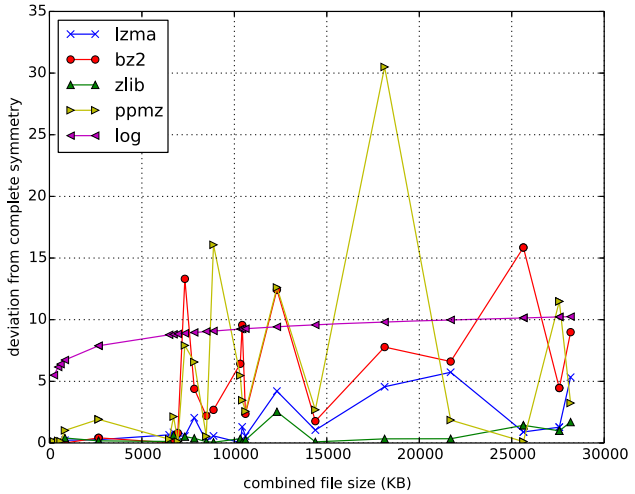

Fig. 3 Symmetry: the difference between $|C(XY)|$ and $|C(YX)|$, as compared to $\log(|XY|)$

Distributivity The difference between $|C(XY)|+|C(Z)|$ and $|C(XZ)| + |C(YZ)|$ is shown in Figs. 4 and 5. As required by the distributivity property, these values are consistently

[1] These are standard corpora for the evaluation of compression algorithms and are available at http://www.data-compression.info/Corpora/.

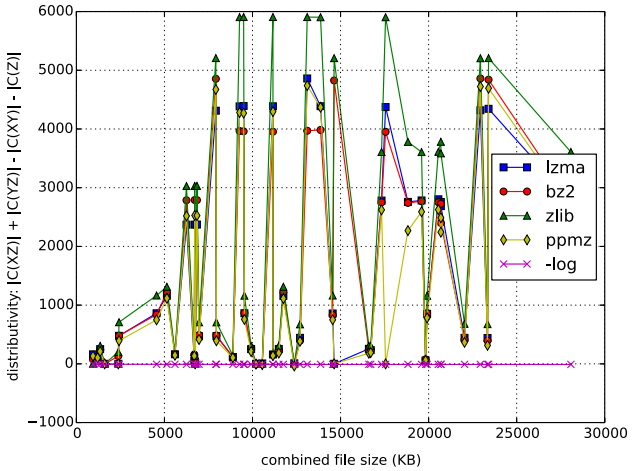

Fig. 4 Distributivity: the difference between $|C(XY)| + |C(Z)|$ and $|C(XZ)| + |C(YZ)|$. If distributivity holds, this value should be non-negative (or at least within $O(\log(n)$ of non-negative

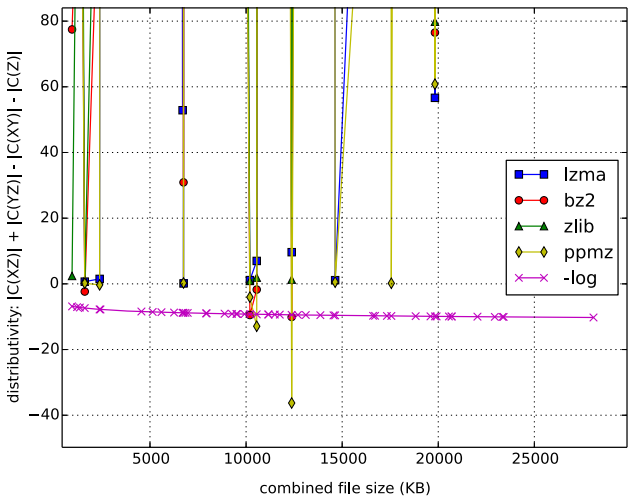

Fig. 5 Distributivity: close-up of the x axis for the difference between $|C(XY)| + |C(Z)|$ and $|C(XZ)| + |C(YZ)|$

non-negative for lzma and zlib. While bz2 and PPMZ go significantly negative in one or two cases, their asymptotic behavior is unclear.

Monotonicity As shown in Fig. 6, all four compressors solidly satisfy the monotonicity property, with $|C(XY)| - |C(X)| > 0$ in all cases.

Our experiments have shown serious violation of the idempotence axiom that has been used to prove theoretical properties of NCD, leaving a potential gap between theory and practice. The next section explores the extent to which NCD can be useful in spite of this gap.

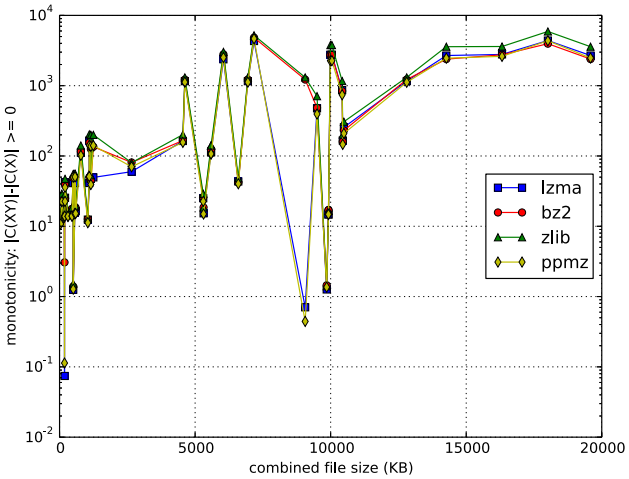

Fig. 6 Monotonicity: $|C(XY)| - |C(X)| \geq 0$

3.2 Classification using NCD with abnormal compressors

We have demonstrated that none of the compression algorithms we explored satisfy the requirements for normal compression. The question remains whether this contraindicates their use with NCD. As mentioned above, much previous work has demonstrated NCD's utility with some of these compression algorithms in applications with small file sizes. However, the compressors' deviation from normality grows with file size. Do they remain useful with larger files?

To address this question, we explored the accuracy of NCD in identifying the malware family of APK files from the Android Malware Genome Project dataset [24,25]. In particular, we took a subset of 500 samples from the Geinimi, DroidKungFu3, DroidKungFu4, and GoldDream families.[2] We used the complete raw APK files, without modification, as our samples. Geinimi samples in this dataset have size up to 14.1 MB, DroidKungFu3 up to 15.4 MB, DroidKungFu4 up to 11.2 MB, and GoldDream up to 6.4 MB.

We evaluated NCD with the same four compression algorithms as above, using a nearest neighbor classifier [8] with a single (randomly selected) instance of each malware family in the reference set.[3] Note that we intentionally restricted the reference set to make the classification problem difficult, in order to explore the limitations of the compression algorithms when used with NCD. Results are shown in Fig. 7. In spite of clearly violating the idempotence property, both lzma and

[2] We selected these families due to their containing enough samples to allow for a meaningful test, and containing large enough files to challenge the compressors.

[3] For readers unfamiliar with nearest neighbor classification, specifically we classified a "test" sample by looking at the distance between it and each of the "reference" samples, and selecting the family of the nearest (i.e. most similar) reference sample.

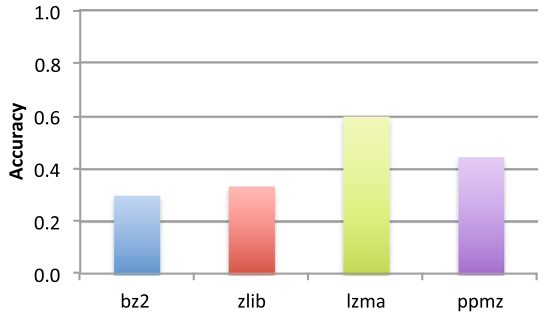

Fig. 7 Accuracy of NCD in identifying Android malware family, using a 1-NN classifier

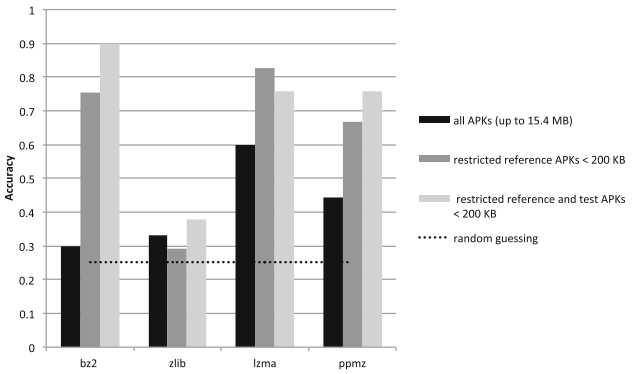

Fig. 8 Effect of file size on accuracy of NCD in identifying Android malware family, using a 1-NN classifier

PPMZ performed significantly better than random guessing. In line with their relative normality, lzma performed best at, 59.7 % with PPMZ up next at 44.4 %. Although bz2 is slightly closer to satisfying the idempotence property than zlib, zlib actually outperformed bz2, albeit not by much, with accuracies of 33.3 and 29.8 %, respectively, with neither performing much better than random guessing.

To demonstrate the relevance of file size, we performed the same test with one slight change, this time using only reference samples smaller than 200 KB. We saw drastic improvement with bz2 (now 75.4 %), lzma (82.5 %), and PPMZ (66.7 %), while zlib's performance actually got worse (29.2 %).

Finally, looking only at files smaller than 200 KB yielded improved performance by bz2 (89.7 %), zlib (37.9 %), and PPMZ (75.9 %), but lzma actually performed slightly worse (75.9 %). The latter suggests that file size is not the only factor that can inhibit the performance of a compression algorithm with NCD. Notably, bz2 outperformed lzma on these files. These results are shown in Fig. 8.

4 Adapting NCD to handle large files

We saw in Sect. 3.2 that NCD has widely varying performance on large files, depending on the compression algorithm used. The memory limitations of the algorithm are key here. The major hurdle is to effectively use information from string X for the compression of string Y in computing $C(XY)$. Algorithms like bz2 and zlib have an explicit block size as a limiting factor; if $|X| >$ block_size, then there is no hope of benefiting from any similarity between X and Y. In contrast, lzma doesn't have a block size limitation, but instead has a finite dictionary size; as it processes its input, the dictionary grows. Once the dictionary is full, it is erased and the algorithm starts with an empty dictionary at whatever point it has reached in its input. Again, if this occurs before reaching the start of Y, hope of detecting any similarity between X and Y is lost. Likewise, even if X is small, but Y is large, with the portion of Y that is similar to X appearing well into Y, the similarity can't be detected.

Thus, it seems logical that we could improve the effectiveness of NCD by bringing similar parts of X and Y in closer proximity of one another; rather than computing NCD using $C(XY)$, we propose using $C(J(X, Y))$ where J is some method of combining strings X and Y. So, we define

$$\mathrm{NCD}_{C,J} = \frac{|C(J(X,Y))| - min(|C(X)|, |C(Y)|)}{max(|C(X)|, |C(Y)|)}.$$

In the original definition of NCD, J is simply concatenation. In an ideal world, J would locate similar chunks of X and Y and place them adjacently. However, if J is too destructive of the original strings, much of the original compression of X and Y individually will be lost, resulting in a higher overall value for $\mathrm{NCD}_{C,J}(X, Y)$. Thus, we want these similar chunks to be as large as possible so as to still allow both chunks to fit within the block size, or to allow processing of them both within the same dictionary. There are some simple ways to achieve this.

One approach would be to apply a string alignment algorithm to X and Y, and combine the two strings so that aligned segments are located in sufficient proximity. However, while Hirschberg's algorithm [13] allows for such alignment to be performed in linear space, thus eliminating memory issues, it takes time proportional to the product of the file sizes and is thus quite slow with large files. Further, this is limited to finding a very specific type of similarity, which is order-dependent. However, we propose two other approaches inspired by this notion.

Interleaving The simplest approach is to assume that similar parts of x and y are similarly located, and just weave them together in chunks of size b. Say $X = x_1x_2, \ldots, x_n$ and $Y = y_1y_2, \ldots, y_m$, where $|x_i| = |y_j| = b$ for $1 \leq i \leq n-1$ and $1 \leq j \leq m-1$, $0 \leq |x_n| < b$, and $0 \leq |y_m| < b$. Then define

$$J_b(x, y) = \begin{cases} x_1y_1x_2y_2 \ldots x_ny_ny_{n+1} \ldots y_m & \text{if } n < m \\ x_1y_1x_2y_2 \ldots x_my_mx_{m+1} \ldots x_n & \text{otherwise} \end{cases}$$

NCD-shuffle Another approach is to split both strings into chunks of the desired size (selected to be appropriate for the compression algorithm) and apply the traditional NCD to determine the similarity of each chunk of X to each chunk of Y, and align them accordingly, with the most similar chunks from the two strings adjacent.

4.1 NCD adaptation results in malware classification

Using the original classification problem from Sect. 3.2, we applied the interleaving (IL) and NCD-shuffle (NS) file combination techniques with various block sizes with each of the compression algorithms. As shown in Table 1 and Fig. 9, in all cases, one or both techniques yielded a better performance than the traditional NCD. Figure 9 also includes the accuracy when 5 representatives from each family are used for comparison (with the exclusion of PPMZ, which was too slow for this experiment). Most notably, these techniques boosted bz2 from 29.8 % accuracy to 52.2 % accuracy with a single training sample, and from 55.2 to 75.2 % with 5 training samples, and boosted zlib from 30 to 74.8 % with 5 training samples.

Table 1 Comparison of performance of different combining functions with NCD in a 1-NN classifier for Android malware family identification, with varying block sizes (block sizes in thousands of KB)

	Concat	IL 1	IL 10	IL 100	IL 1000
bz2	0.298	0.464	0.462	0.456	0.308
zlib	0.333	0.19	0.194	0.131	0.317
lzma	0.597	0.637	**0.643**	0.635	0.603
PPMZ	0.444	0.357	**0.484**	0.438	0.442
	Concat	NS 10	NS 100	NS 1000	
bz2	0.298	**0.522**	0.423	0.325	
zlib	0.333	**0.433**	0.200	0.325	
lzma	0.597	0.641	**0.643**	0.627	
PPMZ	0.444	0.371	0.438	0.435	

Bold values indicate the most accurate of the combining functions compared (for each compression algorithm.)

Table 2 Comparison of performance of different combining functions with NCD in a 1-NN classifier for Windows malware family identification, with varying block sizes (block sizes in thousands of KB)

	Concat	IL 10	IL 100	IL 1000	IL 2000
bz2	**0.922**	0.895	0.895	0.553	0.431
zlib	0.505	0.835	0.519	0.565	0.485
lzma	0.907	0.877	0.871	0.905	**0.909**
PPMZ	0.891	0.893	0.893	0.891	0.883
	Concat	NS 10	NS 100	NS 1000	
bz2	**0.922**	0.887	0.889	0.887	
zlib	0.505	**0.839**	0.704	0.565	
lzma	0.907	0.843	0.911	0.907	
PPMZ	0.891	–	**0.895**	0.891	

Bold values indicate the most accurate of the combining functions compared (for each compression algorithm.)

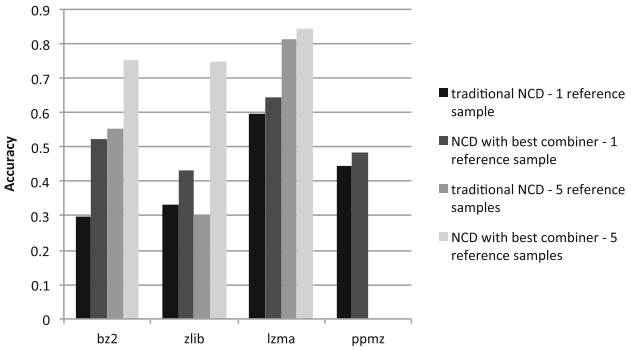

Fig. 9 Traditional NCD compared to the best of the alternative combiners we explored for Android malware family identification

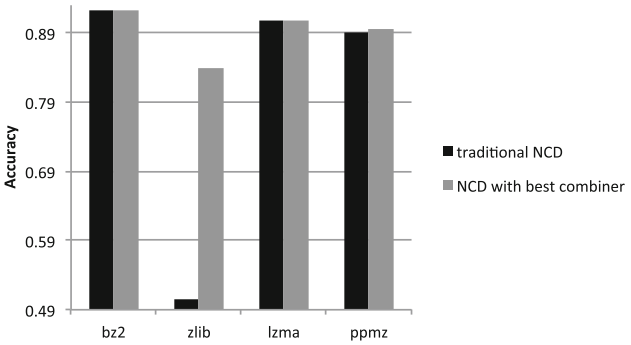

Fig. 10 Traditional NCD compared to the best of the alternative combiners we explored for Windows malware family identification. (Note that the y axis has been truncated to allow small differences to be visible.) While hard to see, slight improvement was shown with lzma and PPMZ. Because these malware files are small, only zlib showed significant improvement

We repeated this experiment with 500 samples from the Lollipop, Kelihos_ver3, and Gatak Windows malware families, from Microsoft's kaggle BIG 2015 Malware Classification Challenge dataset[4] [15]. These samples consisted of Windows binaries with their headers removed. (We did not use the disassembly files that were also included in the data set.) Note that these files, all under 4 MB, are not as large as the Android malware files. As shown in Fig. 10 and Table 2, our techniques boosted zlib from 50.5 % accuracy to 83.9 %, PPMZ from 89.1 to 89.5 %, and lzma from 90.7 to 90.9 %, but offered no improvement with bz2. (Note that the y-axis in Fig. 10 has been truncated in an attempt to allow small differences to be visible.) With the smaller size

[4] Although use of kaggle datasets is normally restricted to the corresponding competition, Microsoft has granted permission for this dataset to be used for academic work.

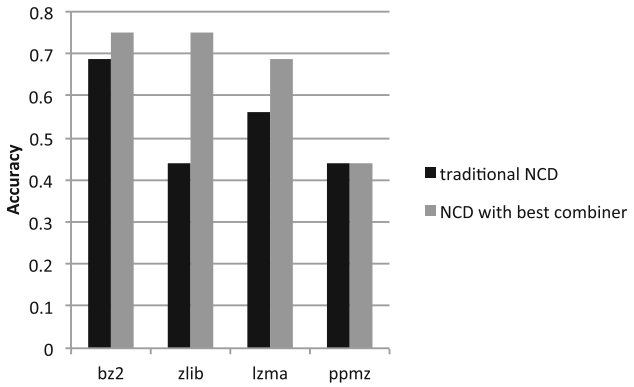

Fig. 11 Traditional NCD compared to the best of the alternative combiners we explored for music artist/composer identification

of these files, and with all but zlib doing reasonably well with the standard NCD to begin with, it is not surprising that these improvements are less dramatic than the results with the larger Android malware files.

Note that we also performed smaller experiments (shown in the Appendix) with these techniques on music and medical image files, and also saw improvements there, so we expect these techniques to offer improvement not just in malware classification, but in all domains where large files are prevalent.

5 Conclusion and future directions

We have demonstrated that several compression algorithms, lzma, bz2, zlib, and PPMZ, apparently fail to satisfy the properties of a normal compressor, and explored the implications of this on their capabilities for classifying malware with NCD. More generally, we have shown that file size is a factor that hampers the performance of NCD with these compression algorithms. Specifically, we found that lzma performs best on this classification task when files are large (at least in the range we explored), but that bz2 performs best when files are sufficiently small. We have also found zlib to generally not be useful for this task. PPMZ, in spite of being the top performer in terms of idempotence, did not come close to the most accurate compressor in any case. Finally, we introduced two simple file combination techniques that improve the performance of NCD on large files with each of these compression algorithms.

However, the challenges of choosing the optimal compression algorithm and the optimal combination technique (and parameters therefor) remain. For supervised classification applications, it is easy enough to use a test set to aid in the selection of the technique and block size parameter for the relevant domain. However, for clustering or genealogy tasks, the burden remains to study several resulting clusterings or hierarchies to determine which is most appropriate.

It remains for future work to better understand what properties of a data set make it more or less amenable to the different compression algorithms and different combination techniques and parameters.

Nonetheless, these techniques offer enhanced NCD performance in malware classification (as well as other tasks) with large files, and suggest that further research in this direction is worth pursuing.

Acknowledgments The author thanks her colleagues at CyberPoint Labs, Mark Raugas, Mike West, Charlie Cabot, James Ulrich, David Ritch, Elizabeth Hughes, and Ian Blumenfeld, for their enthusiastic support and helpful input at various stages of this work.

Appendix: NCD adaptations on music and medical image data

To demonstrate the generality of our approach, we include some NCD improvement results on non-malware files.

We first performed a small experiment, analogous to the ones in Sect. 4.1, in identifying the artist/composer of music MP3 files. We took a set of 20 MP3 files, with sizes ranging from 2 to 10 MB, with the goal of identifying the content as Bach, Handel, Telemann, or the Yes Tones. As before, we randomly selected one reference sample from each artist. Again, we saw significant improvement with our approach. As shown in Fig. 11, zlib improved from 43.8 to 75 % accuracy, bz2 from 68.8 to 75 %, lzma from 56.3 to 68.8 %. (PPMZ showed no improvement.) Interleaving and NCD-shuffle performed comparably on these files.

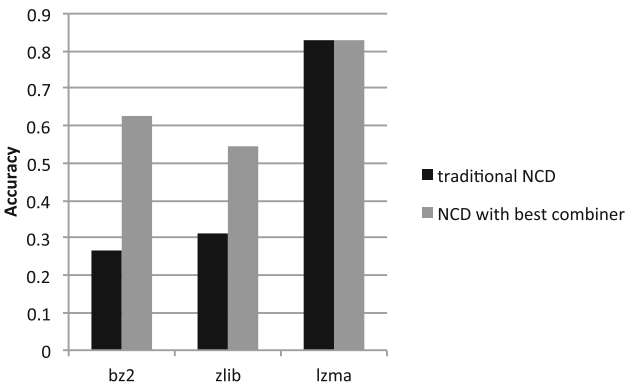

Fig. 12 Traditional NCD compared to the best of the alternative combiners we explored for cancer detection

We then performed a slightly larger experiment with medical image files. We repeated the previous experiment, this time with 68 mammography LJPEG images from DDSM [11,12], with sizes ranging from 6 to 67 MB. The goal was to determine which images were cancerous, and we attempted this using a single reference cancerous image and a single benign one. Results are shown in Fig. 12. There were dramatic improvements with zlib and bz2: zlib improved from 31.3 % accuracy to 54.7 % accuracy in identifying cancerous images, and bz2 improved from 26.6 to 62.5 % accuracy. However, lzma, which achieved a solid 82.5 % with the traditional NCD definition, saw no improvement. (We do not have results for PPMZ, as it was excessively slow on these files.) Interleaving with a block size of 100 KB was the best approach we tried for bz2, while NCD-shuffle with a block size of 10MB was the best for zlib.

While these experiments were small-scale, they provide strong evidence that our NCD adaptations are beneficial in a wide range of classification problems with large file sizes.

References

1. Bailey, M., Oberheide, J., Andersen, J., Mao, Z.M., Jahanian, F., Nazario, J.: Automated classification and analysis of internet malware. In: Recent advances in intrusion detection, pp. 178–197. Springer (2007)
2. Bloom, C.: PPMZ: High compression markov predictive coder. http://www.cbloom.com/src/ppmz.html. Accessed: 2015–04-14
3. Cebrian, M., Alfonseca, M., Ortega, A., et al.: Common pitfalls using the normalized compression distance: what to watch out for in a compressor. Commun. Inf. Syst. **5**(4), 367–384 (2005)
4. Chen, X., Francia, B., Li, M., Mckinnon, B., Seker, A.: Shared information and program plagiarism detection. IEEE Trans. Inf. Theory **50**(7), 1545–1551 (2004)
5. Cilibrasi, R., Cruz, A.L., de Rooij, S., Keijzer, M.: Complearn. http://www.complearn.org. Accessed: 2015–04-15
6. Cilibrasi, R., Vitanyi, P., De Wolf, R.: Algorithmic clustering of music. In: Web Delivering of Music, 2004. WEDELMUSIC 2004. Proceedings of the Fourth International Conference on, pp. 110–117. IEEE (2004)
7. Cilibrasi, R., Vitányi, P.M.: Clustering by compression. IEEE Trans. Inf. Theory **51**(4), 1523–1545 (2005)
8. Cover, T., Hart, P.: Nearest neighbor pattern classification. IEEE Trans. Inf. Theory **13**(1), 21–27 (1967)
9. Dandu, R.V.: Storage media for computers in radiology. Indian J. Radiol. Imaging **18**(4), 287 (2008)
10. Gailly, J.L., Adler, M.: zlib: A massively spiffy yet delicately unobtrusive compression library. http://www.zlib.net. Accessed: 2015–04-14
11. Heath, M., Bowyer, K., Kopans, D., Kegelmeyer Jr, P., Moore, R., Chang, K., Munishkumaran, S.: Current status of the digital database for screening mammography. In: Digital mammography, pp. 457–460. Springer (1998)
12. Heath, M., Bowyer, K., Kopans, D., Moore, R., Kegelmeyer, P.: The digital database for screening mammography. In: Proceedings of the 5th International Workshop on Digital Mammography, pp. 212–218 (2000)
13. Hirschberg, D.S.: A linear space algorithm for computing maximal common subsequences. Commun. ACM **18**(6), 341–343 (1975)
14. Li, M., Chen, X., Li, X., Ma, B., Vitányi, P.M.: The similarity metric. IEEE Trans. Inf. Theory **50**(12), 3250–3264 (2004)
15. Microsoft malware classification challenge (BIG 2015). https://www.kaggle.com/c/malware-classification/data. Accessed: 2015–10-27
16. Pavlov, I.: 7-zip. http://www.7-zip.org. Accessed 14 Apr 2015
17. Seward, J.: bzip2: Home. http://www.bzip.org. Accessed 14 Apr 2015
18. Thom, D., Heidemann, G.: The normalized compression distance as a distance measure in entity identification (2010)
19. Tran, N.: The normalized compression distance and image distinguishability. In: Electronic Imaging 2007, pp. 64,921D–64,921D. International Society for Optics and Photonics (2007)
20. Väyrynen, J.J., Tapiovaara, T., Kettunen, K., Dobrinkat, M.: Normalized compression distance as an automatic mt evaluation metric. Proc. MT **25**, 21–22 (2010)
21. Walenstein, A., Hayes, M., Lakhotia, A.: Phylogenetic comparisons of malware. In: Virus Bulletin Conference, vol. 39, p. 41 (2007)
22. Wehner, S.: Analyzing worms and network traffic using compression. J. Computer Secur. **15**(3), 303–320 (2007)
23. Witten, I., Bell, T., Cleary, J.: The calgary corpus. http://corpus.canterbury.ac.nz/descriptions/#calgary. Accessed 15 Apr 2015
24. Zhou, Y., Jiang, X.: Android malware genome project. http://www.malgenomeproject.org. Accessed 15 Apr 2015
25. Zhou, Y., Jiang, X.: Dissecting android malware: characterization and evolution. In: IEEE Symposium on Security and Privacy (SP), 2012, pp. 95–109. IEEE (2012)